VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

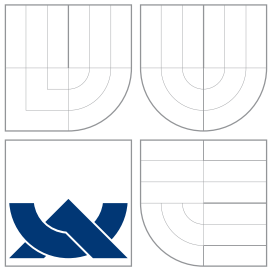
SÍŤOVÁ APLIKACE PRO KARETNÍ HRU MAGIC: THE GATHERING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

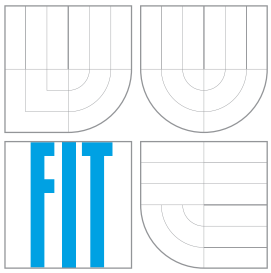
AUTOR PRÁCE
AUTHOR

JAKUB STONAVSKÝ

BRNO 2012



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

SÍŤOVÁ APLIKACE PRO KARETNÍ HRU MAGIC: THE GATHERING

NETWORK APPLICATION OF THE CARD GAME MAGIC: THE GATHERING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — JAKUB STONAVSKÝ
AUTHOR

VEDOUCÍ PRÁCE — Ing. JAROSLAV STRUŽKA
SUPERVISOR

BRNO 2012

Abstrakt

V této práci je popsán návrh a implementace on-line hry dle sběratelské karetní hry Magic: The Gathering. Jedná se o aplikaci typu klient-server implementovánu v jazyce C++ s použitím Qt frameworku. Také jsou v této práci vysvětleny síťové modely ISO/OSI, TCP/IP a model klient-server.

Abstract

This thesis describes the design and implementation of online trading card game Magic: The Gathering. Developed application is client-server based and implemented in C ++ using the Qt framework. Network models ISO / OSI, TCP / IP and client-server based applications are also explained.

Klíčová slova

Magic: The Gathering, síťová aplikace, Qt, C++, karetní sběratelská hra, model klient-server

Keywords

Magic: The Gathering, network application, Qt, C++, collector card game, model client-server

Citace

Jakub Stonavský: Síťová aplikace pro karetní hru Magic: The Gathering, bakalářská práce, Brno, FIT VUT v Brně, 2012

Síťová aplikace pro karetní hru Magic: The Gathering

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Jaroslava Stružky. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jakub Stonavský
13. května 2012

Poděkování

Děkuji vedoucímu Ing. Jaroslavovi Stružkovi za konzultace, ochotu při řešení problémů a pomoc při testování aplikace.

Obsah

Kapitola 1

Úvod

V dnešní době je velký zájem o on-line hraní her. Sběratelská karetní hra Magic: The Gathering (dále jen MTG) se stala taktéž celosvětově populární hrou. Toto jsou přední důvody pro vývoj síťové hry. Pokud se poohlédneme po jiném ztvárnění MTG v on-line hrách, nenalezneme skoro žádné obstojné zpracování, což přináší velmi příjemnou šanci pro velkou oblíbenost budoucí aplikace.

Cílem bakalářské práce je vytvořit síťovou aplikaci týkající se karetní hry MTG. Aplikace by měla přinést intuitivní a jednoduché uživatelské rozhraní, možnost tvorby vlastního karetního balíčku, nahlížení na již vytvořené balíčky, přidávání nových karet a samozřejmě samotnou hru. Aplikace bude podporovat hru dvou hráčů, ale je vyvíjena s ohledem na případná rozšíření. Například: více hráčů hrajících jednu hru, podpora uživatelských účtů a statistik vedených na serveru.

Práce je rozdělena do osmi kapitol. Druhá kapitola seznámí hráče s hrou MTG. Hráči je představena hra, její princip a základní pravidla. Je zde popsán a vysvětlen vzhled karty, její vlastnosti, manová cena, atd. Hráč je obeznámen s fázemi kola, dozví se, jaká kouzla kdy smí hrát, čím je má zaplatit a v neposlední řadě jak probíhají souboje a rozdělení zranění.

Třetí kapitola je zaměřena na teorii týkající se sítí a návrhu algoritmů za pomocí vývojových diagramů. Jsou zde představeny síťové modely ISO/OSI a TCP/IP. Především model OSI má popsán všechny vrstvy, a zároveň vysvětleny jejich významy. Tato kapitola také popisuje rozdíl mezi TCP a UDP přenosy, a problematiku aplikací typu klient a server.

Čtvrtá kapitola se zabývá konceptem aplikace. Hlavně jde o vzhled GUI, vyhodnocování soubojů, hraní kouzel a zpracování stacku. V této kapitole je také popsán návrh souborů, které obsahují informace o balíčku, možnosti jejich zobrazení a tvorby. Poslední část čtvrté kapitoly je zaměřena na návrh komunikačního protokolu mezi klientskou a serverovou aplikací.

Implementací aplikace se zabývá pátá kapitola. V úvodu je představen použitý software pro vývoj aplikace. Zde jsou popsány a vysvětleny signály a sloty sloužící pro komunikaci mezi objekty, obsluha a zachytávání signálu generovaných myší, a také spojení po síti. První část je zaměřena na serverovou aplikaci, a to především na obsluhu klientů, práci s databází, ve které jsou uloženy informace o kartách, rušení a vytváření hry. Ve druhé části páté kapitoly je popsána implementace klientské aplikace. Zde nalezneme implementace herních objektů, změny fází kola, hraní kouzel, zpracování stacku, soubojů bytostí a GUI.

Šestá kapitola seznamuje uživatele s ovládáním a spouštěním aplikací. Jsou zde popsány parametry, které přijímá server při spuštění a jeho chování, pokud není žádný parametr zadán. Z pohledu klientské aplikace se šestá kapitola zaměřuje na vytváření nových her,

připojení k již existující hře a ovládání samotné hry.

Předposlední kapitola se zabývá testováním aplikace. Jsou zde popsány testy, které byly navrženy pro ověření komunikace mezi klientem a serverem, následné testování GUI a průběhu samotné hry.

Poslední kapitola shrnuje celou práci a přináší závěrečný pohled na vývoj aplikace.

Kapitola 2

Magic: The Gathering

MTG je sběratelská karetní hra, která vznikla roku 1993 v USA. Díky veliké podpoře tvůrců se stala velmi brzy celosvětově populární. Úspěch hry je podpořen vznikem turnajového soutěžení o hodnotné ceny. Velkou nevýhodou hraní MTG je cena karet. Pokud chce člověk hrát jen pro zábavu, lze pořídit základní karty za pár korun. Při trošku serioznější hře se však zmíněná zábava stává finančně velmi vyčerpávající. Díky těmto skutečnostem již vznikl jeden významnější projekt, který se zabývá zpracováním on-line hry MTG. Aplikace má implementováno velmi složité a neintuitivní uživatelské rozhraní, díky kterému se stává skoro nehratelnou. Většinu herních pravidel musí hráči řešit samostatně bez jakékoliv kontroly ze strany aplikace. Díky tomu se většinou hry ani nedohrají. Také dochází k neshodám mezi hráči, anebo komunikačním problémům. Pokud hráč neumí výborně anglicky, stává se pro něj hra nehratelnou. Toto jsou nedostatky, kterých jsme se při vývoji aplikace snažili vyvarovat.

Velkým plusem hry je její neustálý vývoj. V pravidelných intervalech jsou vydávány nové edice, které vždy přinesou nějakou lehčí obměnu pravidel, něco do nich přidají či pozmění. Tímto se hra udržuje pořád velmi zábavnou a změnou karet v balíčku či nalezení nového protihráče je každá hra jedinečným a neopakovatelným zážitkem.

2.1 Princip hry

Každý hráč si sestaví svůj vlastní minimálně šedesáti karetní balíček. Velkou výhodou MTG je různorodost herních balíčků. Hráči si je sestavují z rozmanitého množství již vydaných karet. Každý hráč si může nakoupit různé produkty týkající se hry. Především je možné si pořídit starter balíčky, které jsou tvořeny pro nové začínající hráče. Neobsahují nijak zvlášť silné karty, ale představují vždy všechny nové vlastnosti dané edice. Dalším produktem jsou event balíčky. Ty by již měly být obstojnější kvality. Bez úprav mohou obstát na některých menších turnajích. Posledním významným produktem jsou booster balíčky, které vždy obsahují patnáct náhodných karet dané edice. Kromě těchto základních produktů ještě vycházejí různá dárková balení, speciální karty (jako odměny pro hráče). Lze také nakupovat či vyměňovat kusové karty, atd..

Hra se hraje v různých formátech, které udávají, jaké karty smí být použity při hře. Také je možné zúčastnit se speciálních limited turnajů, na které nejsou potřeba žádné vlastní karty. Hráči utvoří přibližně osmičlennou skupinu, která si postupně rozdraftuje booster balíčky.

Rozdraftování booster balíčku je postup, při němž skupina hráčů (ideální počet je osm

hráčů) obdrží 3 boostery na osobu. Na povel každý hráč otevře jeden booster balíček, prohlídne si karty a jednu si vybere. Pak balíček pošle hráči po své levici a od souseda napravo dostane další booster. Opět vybere jednu kartu a zase pošle balíček hráči po své levici. Po rozebrání prvního boosteru dostanou hráči čas na prohlédnutí svých karet. Pak se otevřou druhé boostery a opakuje se předchozí postup, ovšem s jednou obměnou a tou je změna směru posílání balíčku. Toto se provede i s třetím boosterem. Pak mají hráči čas na poskládání herního balíčku. Podmínkou je minimum 40 karet na balíček. Naopak není omezen počet kusů jedné karty. Po dokončení draftovací fáze započne turnaj mezi hráči.

Pro stavbu herního balíku existují tato omezení: musí obsahovat maximálně 4 kusy od jedné karty (pravidlo může být porušeno, pokud to určí samotná karta) a nesmí obsahovat zakázané karty. Omezení čtyř karet v balíčku neplatí pro základní země, které produkují manu potřebnou pro hraní kouzel. Těchto zemí může být libovolné množství.

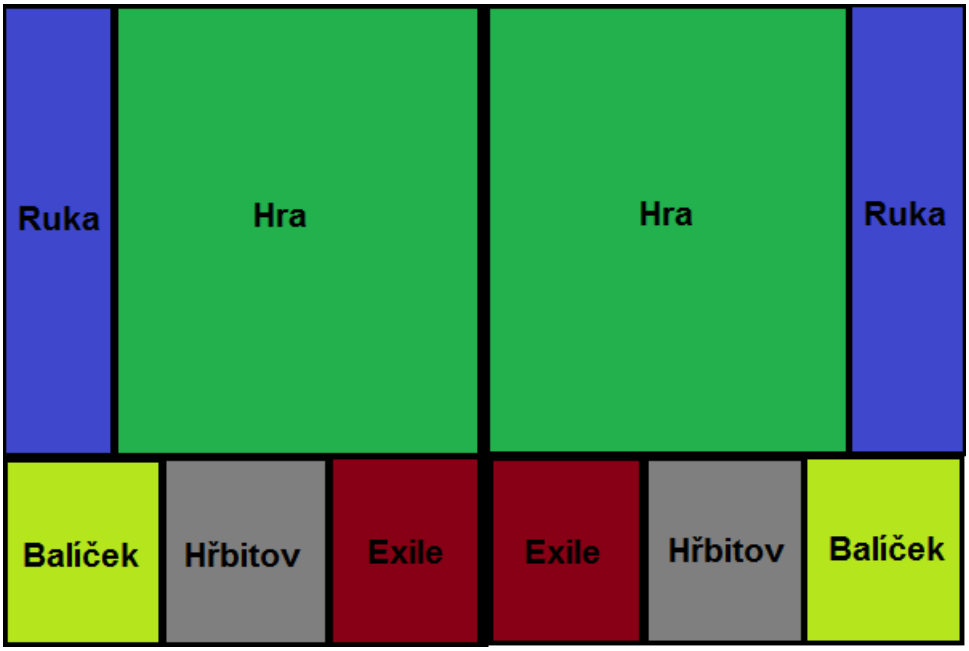
Hráči se v této hře stávají silnými čaroději, kteří mezi sebou soupeří o moc. Balíček karet představuje hráčovu mysl plnou kouzel, s jejíž pomocí vyvolává příšery a kouzlí. Hráči se snaží navzájem zabít, a to buď snížením životů z 20 na 0 nebo vyčerpáním knihovny.

2.2 Pravidla hry

Herní pravidla jsou nadmíru obsáhlá, proto zde probereme pouze jejich základy. Podrobnější informace jsou k nalezení na webových stránkách výrobce hry Wizards of the Coast[8].

2.2.1 Základní schéma hry

Herní plocha se skládá z několika částí viz. obrázek 2.1. Toto schéma zobrazuje rozložení hry pro dva hráče.



Obrázek 2.1: Herní plocha pro dva hráče.

- Ruka – hráčovy karty, které má v ruce. Tyto karty vidí jen daný hráč. Oponent smí nahlédnout do protihráčovy ruky, pokud mu to umožní nějaké kouzlo.

- Hra – do této části herní plochy se vykládají základní země, bytosti, artefakty a enchantmenty. Karty jsou viditelné pro oba hráče.
- Balíček – hráčův balíček karet. Před každou hrou musí být zamíchán. Nevidí do něj žádný hráč.
- Hřbitov – sem přicházejí karty jenž opouštějí hru. Většinou to jsou bytosti, které zničí oponent kouzlem nebo zemřou v souboji bytostí. Dále pak zničené artefakty, enchantmenty, země, a také sorcery či instanty, které mají okamžitý dopad na hru.
- Exile – tato část obsahuje karty, které byly vyřazeny ze hry, a to buď za pomoci kouzel nebo vlastností, které určují, že karta opustí hru a je přesunuta do exilu.

2.2.2 Barvy kouzel a mana

MTG obsahuje pět barev kouzel, ovlivňujících styl hry. Máme zde barvu bílou (léčení-ochrana vlastních karet), černou (zabíjení bytostí, oživování vlastních karet), červenou (působení zranění, rychlé malé bytosti), modrou (kontrola hry) a zelenou (růst síly, silné levné bytosti). Každá barva má svůj typ základních zemí. Jsou to pláně (plains), bažiny (swamp), hory (mountain), ostrovy (island) a lesy (forest). Ty dokáži jednou za hráčovo kolo vyprodukovat manu, která se použije k zahrání kouzla. Pokud není na konci kola všechna vyprodukovaná mana spotřebována, tak zaniká. Vždy na začátku hráčova kola dojde k obnově jeho zemí, které jsou znovu připraveny k použití pro vytvoření many. Každá karta potřebuje pro seslání různý počet many. Takže silnější kouzla dokáže hráč zahrát až v pozdější hře. Země lze vykládat na hrací stůl jen jednou za každé své kolo. Toto pravidlo může být opět porušeno nějakým kouzlem, které dovolí vyložení další země v témže kole.

2.2.3 Popis karty

- Název – jedinečné pojmenování karty. Karty se stejným názvem se mohou lišit pouze obrázkem a edicí.
- Manová cena – udává, kolik many musíme zaplatit, pokud chceme zahrát dané kouzlo. Určuje nám barvu karty, pokud ji neurčí text na kartě. Většinou se skládá ze značek dané země a čísla v šedém kolečku. Tím je určen celkový počet potřebné many, z nichž některé musí být daného typu. A ostatní mohou byt vyprodukovány jakýmikoliv zeměmi.
- Typ – podtyp – typové a podtypové označení karty. Důležité z pohledu ostatních karet, které mohou ovlivňovat karty určitých typů. MTG obsahuje 6 základních typů karet: creature (bytost), enchantment, instant, sorcery, artifact (artefakt), planeswalker.
- Edice – označení edice, v které karta vyšla. Barva tohoto označení určuje vzácnost karty. Dnes jsou 4 typy: černé (obyčejná karta), stříbrné (neobvyklá karta), zlaté (vzácná karta), červené (mýtická karta).
- Aktivační ability – vlastnosti karty, za jejichž použití musíme zaplatit nějakou cenu. Zápis těchto schopností vypadá následovně: <$\text{cena}_1, \text{cena}_2, \ldots, \text{cena}_n$>: <popis efektu, který se provede>. Tyto schopnosti lze hrát stejně jako instanty, to znamená kdykoliv během hry.

Obrázek 2.2: Herní karta.

- Pasivní ability – vlastnosti karty jež platí stále nebo se spouští v určitém momentě hry. Tyto ability jsou popsány textem, který vysvětlí co se má stát. Také existují ability, které jsou popsány pouze jedním slovem. Tyto ability jsou známy již delší dobu a z důvodu úspory místa na kartách byly označeny jednoznačným názvem.
- Síla/životy – udává se jen u bytostí a určuje, kolik zranění dokáže předat jiné bytosti či hráči a kolik sama vydrží.

Nejpoužívanější jednoslovné ability podporované aplikací:

- Flying – bytost označena touto vlastností, smí být blokovaná pouze bytostí s vlastností flying nebo reach.
- Reach – bytost smí blokovat útočníka, který má vlastnost flying.
- Vigilance – bytost se nemusí tapovat do útoku.
- First strike – bytost má přednost udělit zranění v souboji. Pokud tuto vlastnost mají obě bytosti, udělí si ho klasickým způsobem, a to vzájemně.
- Double strike – bytost udělí zranění First strike + klasické zranění.
- Trample – bytost, která byla blokovaná, udělí do hráče zbytek zranění, které nebylo spotřebováno na zničení blokujících bytostí.
- Flash – kartu je možné hrát stejně jako by byla typu instant.
- Hexproof – kartu nemohou cílit oponentova kouzla.
- Defender – bytost s touto vlastností nemůže útočit.

- Intimidate – bytost může být blokovaná pouze artefaktovými bytostmi nebo bytostmi sdílející s útočící bytostí barvu.
- Lifelink – kolik udělí bytost zranění, tolik si hráč přidá životů.
- Haste – bytost smí být tapnuta hned po příchodu do hry.
- Deathtouch – když bytost s touto vlastností udělí zranění druhé bytosti, je druhá bytost zničena.

2.2.4 Fáze kola a průběh hry

Každé hráčovo kolo se skládá z dvanácti fází, které dovolují provádět různé herní akce.

1. Untap – dojde k obnovení zemí a bytostí unavených útokem. A také bytosti, které projdou touto fází mohou být tapovány.
2. Upkeep – zde se provedou akce, které určují karty vyložené na stole a hráči dostanou šanci zahrát instanty a ability před zahájením hlavní fáze.
3. Draw – hráč si dolízne jednu kartu.
4. Main – v této fázi se mohou hrát všechna kouzla např. vyložit příšeru, zahrát sorcery, atd. Také může být vyložena země potřebná k produkci many (jedna za kolo).
5. Begin combat – fáze před útokem, opět k zahrání případných instantů či abilit.
6. Declare attackers – hráč určí bytosti, které se zúčastní útoku na protivníka. Vždy se útočí na hráče a nebo planeswalkera (další hráč reprezentovaný kartou), nikdy ne na bytosti.
7. Declare blockers – bránící se hráč přidělí k útočícím bytostem blokující bytosti. Každá blokující bytost smí blokovat jen jednoho útočníka. To neplatí opačně, kde více bytostí může blokovat jediného útočníka. Díky tomu dokáží slabší bytosti zneškodnit jednu silnější.
8. Combat damage – dojde k rozdělení bojového zranění. Bytosti, které nebyly blokovány, udělí zranění do hráče a ostatní si jej vymění mezi sebou. Pokud byl jeden útočník blokován více bytostmi, zranění rozděluje hráč, jenž útočil. Bytosti, kterým klesnou životy na 0, opouští hru. Ostatním bytostem zůstává zranění do konce kola.
9. End combat
10. Main – opět stejné možnosti jako ve 4 fázi hry.
11. End – poslední možnost na zahrání instantů a aktivačních abilit.
12. Clean up – uklízení po kole. Dojde k odstranění zranění u bytostí, zahodí se vyprodukované many, které nebyly použity a upraví se počet karet v ruce hráče. Po skončení tahu nesmí mít hráč více jak 7 karet. Pokud tomu tak je, hráč si sám vybere karty, které se mu nelíbí a zahodí je.

V každé fázi kola se smí hrát pouze kouzla typu instant. Pokud chce hráč hrát jiný typ kouzla, musí být splněny tři podmínky: musí být kolo hráče, který chce kouzlo hrát, dále je potřeba, aby byla main fáze kola a nebylo žádné kouzlo na stacku.

Na začátku hry si každý hráč lízne 7 karet ze svého balíčku. Pokud mu nevyhovují, smí je zamíchat zpět a vzít si o jednu méně. Tento postup může hráč opakovat, kolikrát uzná za vhodné. Následně se vybere začínající hráč, který započne hru od čtvrté fáze a tím přeskočí dolíznutí jedné karty. Po průchodu všech fází kola je na tahu protihráč, který již začíná od první fáze.

Hra končí, pokud klesnou jednomu hráči životy na 0, anebo dojde k situaci, že si má líznout kartu a žádná již není v knihovně, anebo některá karta oznámí, že jeden hráč prohrál či vyhrál.

2.2.5 Tapování a karty bytostí

Tapnutí karty je pojem, který udává, že byla karta toto kolo použita za nějakým účelem. Především se tapují země, které již vyprodukovaly manu, bytosti chystající se útočit, anebo karty, které mají tapnutí jako cenu aktivační ability. Pokud dojde k tapnutí karty je otočena o 90°. Tímto způsobem je viditelně označena a již nebude do její obnovy použita.

Karty bytostí po vyložení na stůl nemohou být v daném kole tapnuty. Karta trpí vyvolávací únavou, která ji zakazuje útočit či využívat ability, které mají v ceně symbol tapnutí. Toto pravidlo neplatí pro bytosti, které mají vlastnost haste.

2.2.6 Stack

Stack je reprezentován jako klasický zásobník, kde se na jeho vrchol vkládají nové karty, které byly zahrány. A vždy, když se ze zásobníku odebírá karta, je to ta na jeho vrcholu.

Pokaždé když dojde k zahraní kouzla, tak se vloží na stack. Následně dostanou hráči možnost na tuto situaci reagovat, ale to již jen zahráním instantů nebo aktivační schopnosti některé karty ve hře. Pokud již nikdo dále nereaguje, začnou se karty postupně vybírat a provádět jejich vlastnosti, a to v opačném pořadí než byly hrány.

Kapitola 3

Teorie

Tato kapitola se zabývá teorií počítačových sítí. Bude zde vysvětlen pojem počítačová síť, model klient-server, ale také popsány protokoly ISO/OSI a TCP/IP. V této kapitole bude vysvětlen rozdíl mezi TCP a UDP přenosy po síti. V závěru kapitoly se zaměříme na vývojové diagramy.

3.1 Počítačová síť

Pojmem počítačová síť je označeno propojení několika počítačů, jež spolu následně komunikují. Zasílají si zprávy. Každý počítač je v síti identifikován pomocí IP adresy a přenosy na síti vždy probíhají dle nějakého protokolu. Vyvíjená aplikace je postavena na protokolu TCP.

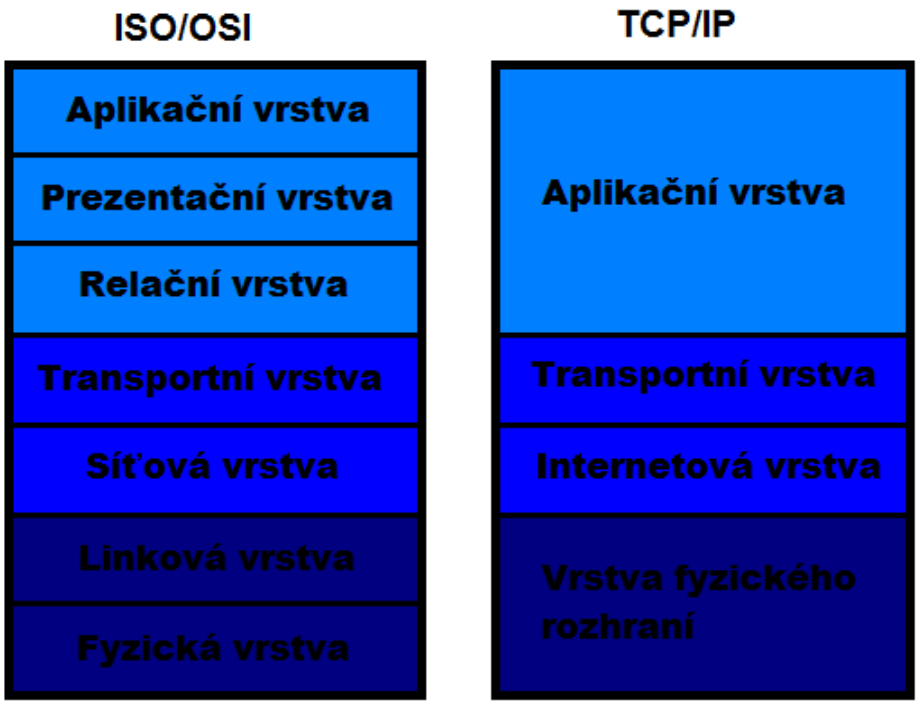
3.2 ISO/OSI

ISO/OSI je vrstvový síťový model, který je využit pro popis síťové architektury. Tento model nebyl nikdy plně aplikován, ale stal se referenčním pro ostatní modely. OSI model se skládá ze sedmi vrstev, kde každá vrstva má definované služby a protokoly, jež na ní pracují. Vždy spolu komunikují dvě sousední vrstvy, a to pouze přes služby, které každá vrstva nabízí. Model ISO/OSI je zobrazen na obrázku 3.1.

- Fyzická vrstva – jsou zde definovány fyzické vlastnosti linky, bez ohledu na to, co data reprezentují. Je zde popsáno, jak se reprezentují logické hodnoty 1 a 0 , jaké jsou použity konektory, síťové kabely, atd.
- Linková vrstva – zde se tvoří rámce, které zapouzdřují data. Také popisuje přenos dat,adresování a zabezpečení proti chybám při přenosu.
- Síťová vrstva – dochází zde k směrování a adresování v síti. Cesta k cíli je tvořena dvěma způsoby, buď dynamicky při průchodu pakety, anebo je stanovena dopředu. Na této vrstvě pracuje protokol IP.
- Transportní vrstva – zajišťuje přenos dat. Přijímá data z vyšší vrstvy a vytváří pakety. Musí být zajištěna kvalita přenosu. Tu definují a požadují vyšší vrstvy. Na této vrstvě jsou protokoly TCP a UDP.
- Relační vrstva – dochází zde k udržování relace, synchronizaci, obnovení spojení mezi dvěmi komunikujícími systémy.

- Prezentační vrstva – je zde zajištěno zpracování dat mezi různými aplikacemi. Data jsou transformována do tvaru, kterému dané aplikace rozumí.
- Aplikační vrstva – s touto vrstvou spolupracují samotné programy. Mají zde rozhraní pro komunikaci po síti.

Informace o modelu ISO/OSI jsou čerpány z [6].



Obrázek 3.1: Síťový model ISO/OSI a TCP/IP.

3.3 TCP/IP

Model TCP/IP vychází z OSI modelu a zjednodušuje jej. Některé vrstvy jsou sjednoceny v jednu vrstvu. Díky tomu je model tvořen pouze čtyřmi vrstvami viz.obrázek 3.1. Předpokládá se, že na nižších vrstvách jsou nespolehlivé služby, takže kontrolu provádějí vždy vyšší vrstvy, a to pouze na vyžádání.

3.4 TCP vs. UDP

Protokol TCP zajišťuje spolehlivý přenos dat po síti. Vždy, když jsou připravena data pro přenos, jsou zapouzdřena do paketů a ty jsou očíslovány. Díky tomuto mohou být později zpětně seřazeny a data jsou opět vytvořena bez chyb. Pokud by došlo ke ztrátě paketu, je vyžádáno o nové zaslání. Nevýhodou tohoto přístupu je jeho rychlost, která je díky větší režii omezena. Například, když je vytížena linka, může docházet k častým ztrátám paketů, které musí být znovu odeslány. Tento protokol je využit vždy, když je potřeba, aby byla doručena všechna data i přes možnost zpomalení přenosu. TCP je použito i v karetní aplikaci MTG, kde nemůžeme dopustit, aby docházelo ke ztrátám dat, která ovlivňují hru.

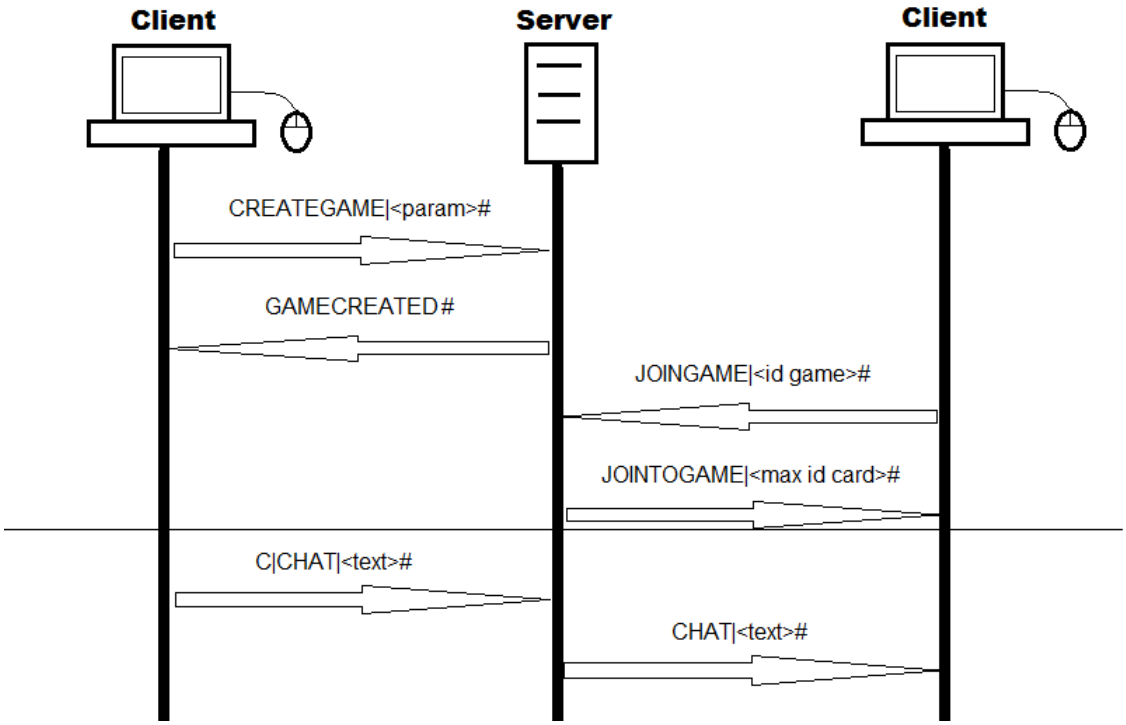
Protokol UDP je oproti předchozímu rychlejší, ale to na úkor spolehlivosti. Tomuto protokolu odpadá režie s navazováním spojení a následného číslování paketů. O toto všechno se musí starat cílová stanice. Takže může docházet k výpadkům nebo přehození paketů. Tento přístup je využit především u streamováni videa, zvuku a akčních her, kdy nevadí občasná ztráta dat.[3]

3.5 Síťový model klient-server

Jedná se o dvě aplikace, které spolu komunikují po síti. Klientská aplikace je většinou ovládána uživatelem a je tvořena grafickým rozhraním pro jednoduchou obsluhu. Pokud klient potřebuje nějakou službu serveru, je vygenerován daný požadavek a odeslán na server. Ten klientův požadavek zpracuje a odešle odpověď opět klientovi, který o ni zažádal.

Serverové aplikace jsou většinou bez grafického rozhraní a běží v nekonečné smyčce. Servery by měly umožnit souběžný přístup několika klientů a zvládnout zpracovat jejich požadavky paralelně. Tento přístup je umožněn více vláknovým programováním nebo tvorbou nových procesů, které vyřeší daný požadavek a server může zatím čekat na další dotazy.

Pro komunikaci mezi serverem a klientem je nutné vytvořit komunikační protokol. Existují standardní servery, ke kterým musí být vytvořena klientská aplikace, která je schopna komunikovat jeho protokolem a opačně. Ale pokud aplikace serveru i klienta vznikají souběžně, je možné vytvořit nový protokol. Ten bude vyhovovat přesným požadavkům daných aplikací. Na obrázku 3.2 je vidět zjednodušená ukázka komunikace dvou klientů se serverem.[1]



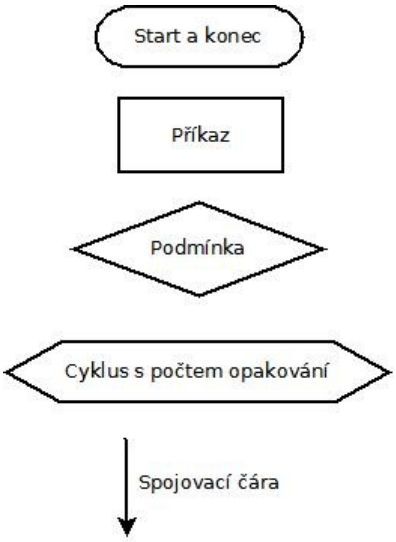
Obrázek 3.2: Komunikace mezi klienty a serverem.

První klient zasílá serveru informace s požadavkem o vytvoření hry a server odpoví, zda

byla hra vytvořena nebo došlo k chybě. Pak druhý klient odešle požadavek s žádostí o připojení ke hře. Pokud je připojení možné, dostane odpověď JOINTOGAME, kde je parametrem zprávy prozatímní maximální hodnota id karet. Klient si pak podle něj očísluje svoje karty. Dále je na obrázku pod vodorovnou čarou ukázána komunikace během hry. Pokud chce jeden klient zaslat zprávu druhému, je vždy zpráva uvozena příkazem C|<zpráva>. Server odebere příkaz C, který určuje, že má být předáno sdělení druhému hráči. Pak již odešle samotnou zprávu, kterou druhý klient zpracuje.

3.6 Vývojové diagramy

Vývojové diagramy slouží ke grafickému popisu a návrhu algoritmů. Každý vývojový diagram se skládá z různých grafických symbolů. Přehled základní symbolů je znázorněn na obrázku 3.3. Každý symbol má určen nějaký význam a funkčnost. Diagramy vznikají spojováním těchto symbolů za pomocí šipek. Cesty tvořené šipkami se nesmí nikdy křížit a vždy musí jít jednoznačně určit, kterou cestou se vydat. Informace o vývojových digramech jsou přejaty z [7] a [5].



Obrázek 3.3: Základní symboly vývojového diagramu.

Kapitola 4

Návrh aplikace

Při návrhu jsme se zaměřili především na vzhled GUI během hry. Je potřeba navrhnout intuitivní rozložení obrazovky, aby bylo možné zobrazit všechny důležité objekty ve hře. Další důležité částí, které je nutné navrhnout jsou souboje bytostí, komunikační protokol, tvorba balíčku a přidávaní nových karet.

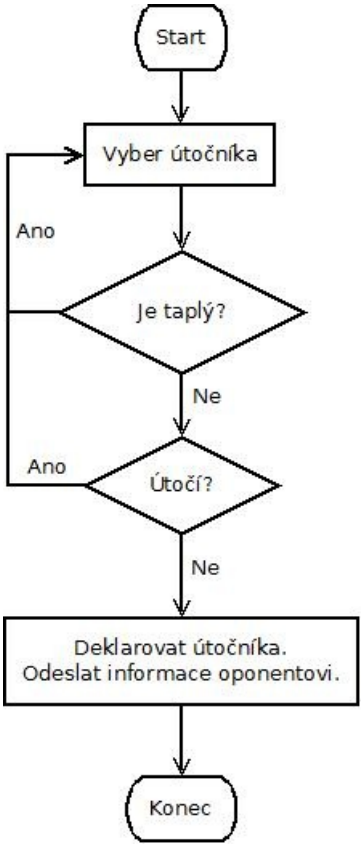
4.1 GUI ve hře

GUI je navrhnuto s ohledem na velké množství informací, které je potřeba zobrazit. Návrh obrazovky je vidět na obrázku 4.1. Ta je rozdělena do dvou hlavních celků. První část zobrazuje informace o hráčích, a to především životy, stav knihovny a počet karet v ruce. Také se zde nachází zobrazení fáze, ve které se zrovna hra nachází. V neposlední řadě zde máme chat pro komunikaci hráčů, který lze skrýt a zobrazit místo něj stack, který zobrazuje právě hraná kouzla. Tímto se uspoří místo, které bude využito pro zobrazení herní plochy. Druhou částí obrazovky je herní plocha. V té musí být vykresleny karty hráče, které má v ruce. Pak se zde nachází karty, které jsou ve hře (bytosti, země, artefakty a enchantmenty). Také zde najdeme hřbitov a exile, do kterých smí hráči nahlížet. Pokud je potřeba zobrazit hřbitov, dojde k zobrazení v novém okně. Ve spodním pravém rohu herní plochy se nachází ovládací tlačítka.

Obrázek 4.1: Návrh GUI ve hře.

4.2 Návrh soubojů

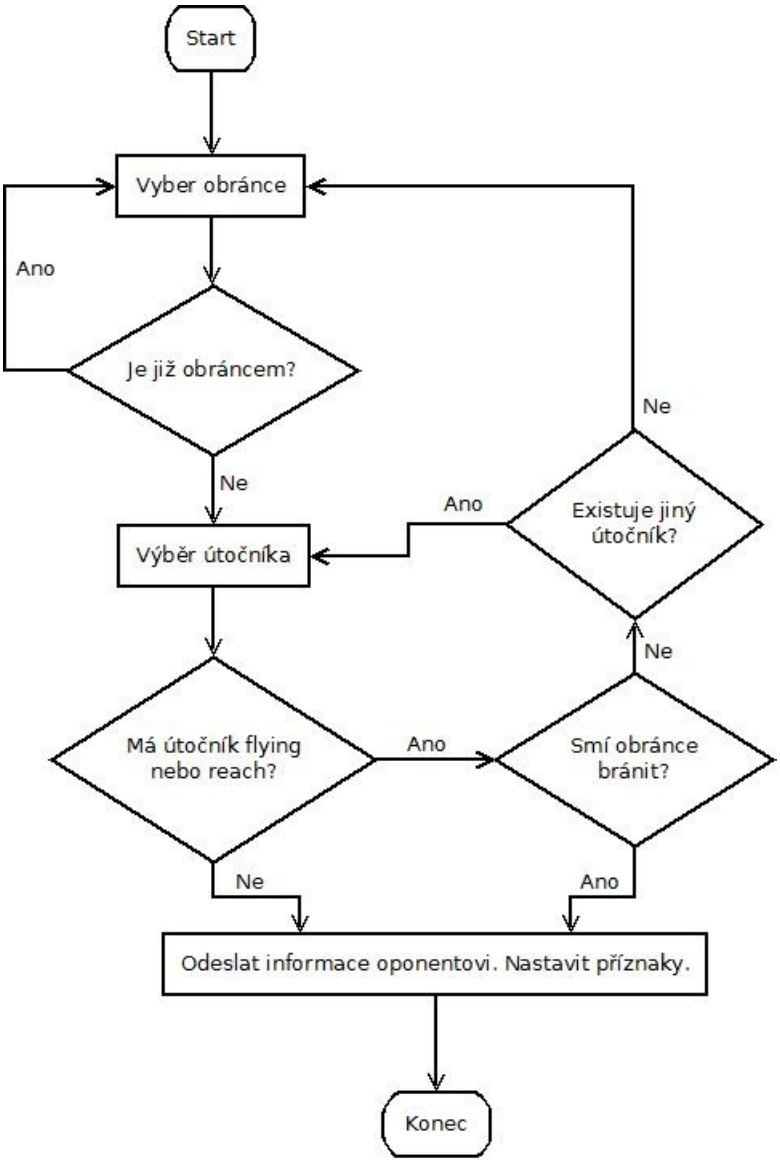
Souboje se skládají ze tří fázi. V první bude útočící hráč označovat útočníky. Toto bude provedeno pouze kliknutím na kartu bytosti. Dojde k otestování, zda může bytost útočit a pokud ano, je označena za útočníka. Tento postup je zobrazen na obrázku 4.2.



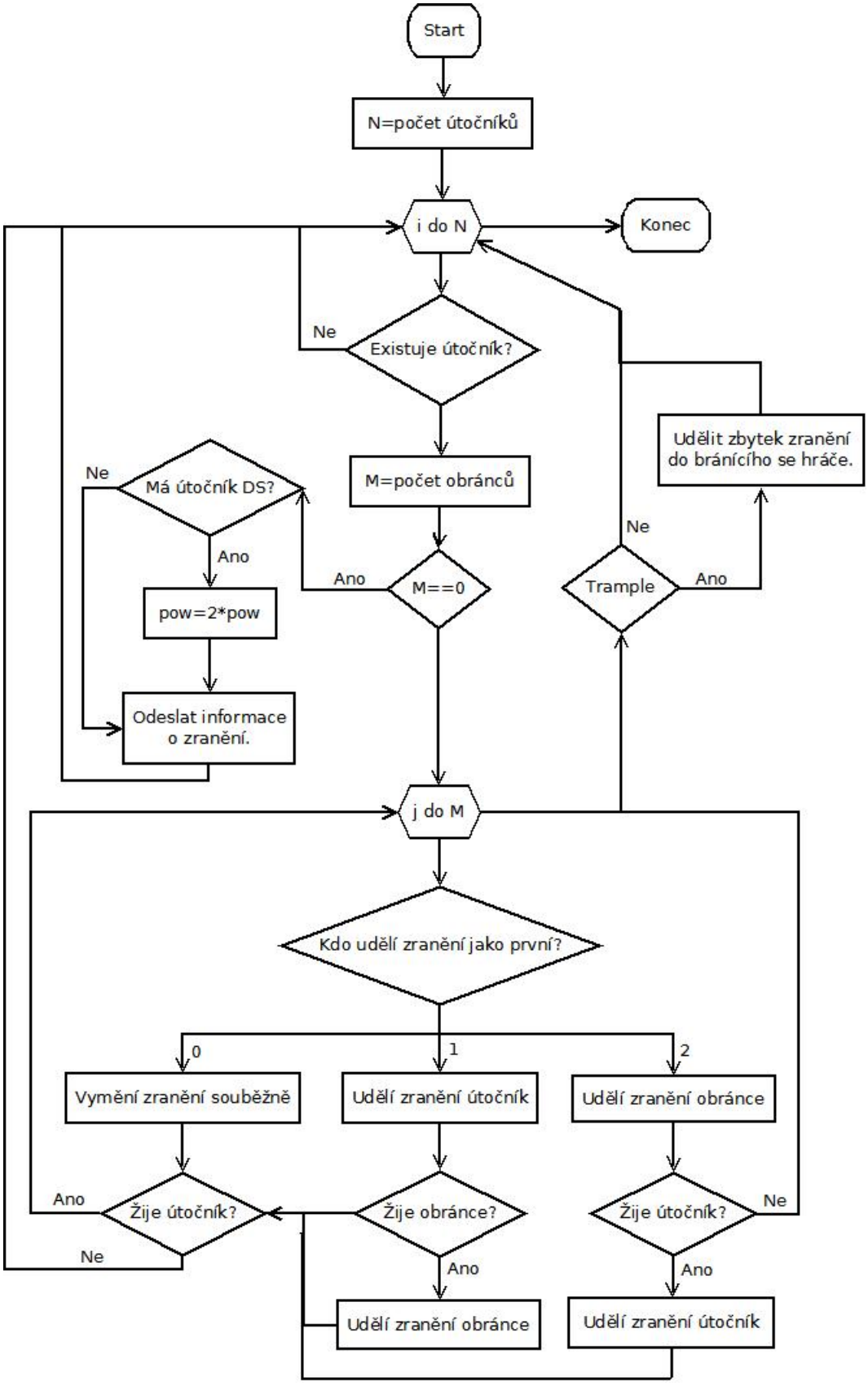
Obrázek 4.2: Deklarace útočníka.

V druhé fázi proběhne deklarace obránců. Algoritmus řešící tento problém je znázorněn na obrázku 4.3.

Blokující hráč vybere obránce, který ještě neblokuje žádnou bytost. Poté musí vybrat útočníka, který bude danou bytostí blokován. Pokud je vybrán obránce i útočník, ověří se, zda může daný obránce blokovat útočníka. Pokud ano, dojde k přiřazení obránce, pokud ne, hráč musí zvolit nového obránce či útočníka. V třetí fázi soubojů dojde k rozdělení zranění. Každá útočící bytost musí udělit zranění, a to buď do hráče nebo do bytostí, které ji blokují. Rozdělení zranění bude probíhat v cyklu, kdy je vždy ze seznamu útočníků jeden vybrán a ten udělí zranění. Pokud nebyl nikým blokován, je zranění uděleno do hráče a dojde k výběru dalšího útočníka. V druhém případě je bytost blokována minimálně jedním obráncem. Pokud k tomuto dojde, tak se v druhém cyklu budou procházet obránci a vymění si zranění s útočníkem. Cyklus je ukončen, pokud již není žádný obránce, který ještě neudělil zranění, anebo zemře útočící bytost. Návrh algoritmu rozdělení bojového zranění je na obrázku 4.4.



Obrázek 4.3: Deklarace obránce.



Obrázek 4.4: Rozdělení bojového zranění.

4.3 Tvorba a zobrazení balíčku

Balíček lze vytvořit ručně, a to editací souboru, který obsahuje seznam karet v balíčku nebo pomocí jednoduchého grafického editoru, který umožňuje vyhledávat a přidávat nové karty do balíčku.

Každý řádek souboru musí začínat číslem, následovaným mezerou a názvem karty. Karty mohou být maximálně čtyři, ale toto neplatí pro základní země (plains, island, forest, swamp, mountain). Soubor může obsahovat komentáře uvozeny znakem # a ukončeny koncem řádku. Zde je ukázka, jak může vypadat případný soubor s informacemi o kartách v balíčku:

#komentare
4 Alabaster Mage
4 Benalish Veteran
4 Demystify
2 Frost Titan
2 Sun Titan
4 Grand Abolisher
4 Peregrine Griffin
4 Personal Sanctuary
4 Phantasmal Bear
4 Sphinx of Uthuun
4 Phantasmal Dragon

11 Plains
11 Island

4.4 Přidávání nových karet

Nové karty smí přidávat pouze uživatel, který zná přístupové informace k databázi karet. Pokud uživatel vytvoří kartu, musí k ní přidat obrázek na straně všech klientů, kteří tuto kartu budou používat při hře. Bez úpravy kódu aplikace, lze přidávat karty bytostí bez složitějších abilit. Karta smí obsahovat všechny jednoslovné ability, které aplikace podporuje.

4.5 Komunikační protokol

První verze komunikačního protokolu obsahovala zprávy ve tvaru <příkaz> <parametr$_1$> ... <parametr$_n$ >. Příkaz a jeho parametry byly odděleny mezerami. Toto se neosvědčilo, protože většina parametrů ve zprávách obsahuje mezery. Druhá verze, již používá jako oddělovač znak |. Zprávy tedy byly v tomto upraveném formátu <příkaz> | <parametr$_1$> | ... | <parametr$_n$>. Změnou oddělovače odpadl problém s mezerami v textu parametrů.

Během testování komunikace mezi klientem a serverem docházelo k slučování zpráv, kdy příjemce nestíhal zprávu přečíst a předat ke zpracování, než byla doručena další. Z tohoto důvodu je na konec zprávy přidán znak #, který odděluje zprávy. Zpracování zpráv bude provedeno ve smyčce, ve které bude vždy vytvořen seznam zpráv ke zpracování. Zprávy pro komunikaci tedy budou následujícího tvaru <příkaz> | <parametr$_1$> |

... | <parametr$_n$>#. Příkaz je vždy textový řetězec pouze z velkých písmen. Parametry již mohou obsahovat všechny znaky mimo oddělovacích znaků | a #. Tento typ zpráv bude využit především v komunikaci klient-server. Během hry budou většinou klienti chtít komunikovat mezi sebou, a proto je navrhnut typ zprávy, který obsahuje dva příkazy. Pokud bude takováto zpráva doručena na server, dojde k jejímu přeposlání druhému klientovi, který je ve hře. Zpráva bude formátu C| <příkaz pro klienta> | <parametr$_1$> | ... | <parametr$_n$>#. Před odesláním zprávy druhému klientovi, dojde k odstranění prvního příkazu a je odeslána klasická zpráva.

Speciálním typem zpráv budou ty, které ponesou informaci o samotných kartách. Klient odešle na server seznam karet a na straně serveru bude vytvořená zpráva <příkaz> | <karta$_1$> |...| <karta$_n$>#. V těchto zprávách bude každá karta obsahovat informace oddělené znakem $. Samotná karta vypadá následovně <informace$_1$>$ ... $<informace$_n$>.

Kapitola 5

Implementace

Aplikace je implementována v programovacím jazyce C++ za použití Qt frameworku vyvíjeného společností Nokia. Qt obsahuje program pro tvorbu grafických aplikací Qt Creator. Za pomocí tohoto nástroje byly vytvořeny základní obrazovky, které aplikace využívá. Tento přístup velmi zjednodušil návrh obrazovek, kdy je neustále vidět, jak vše vypadá a odpadá opakované spouštění aplikace a následné úpravy vzhledu. Při programování samotné hry se již Qt Creator používá jen k překladu a snadnějšímu doplňování kódu, kdy velmi příjemně napovídá části tříd, proměnné, atd..

Qt využívá komunikace mezi třídami, a to za pomocí signálu a slotu. To je využito i u tříd `QTcpServer` a `QTcpSocket`. Díky této komunikaci vždy jen připojíme dva objekty, kde jeden vystupuje jako odesílatel signálu a druhý jako jeho příjemce. Připojení se provádí pomocí funkce `bool QObject::connect ( const QObject * sender, const char * signal, const QObject * receiver, const char * method, Qt::ConnectionType type = Qt::AutoConnection )`. Signály mohou přijímat parametry, které se následně předají danému slotu. Toto je využito v klientské aplikaci. Aby bylo možno se signály pracovat, je nutno do každé třídy přidat makro `Q_OBJECT`.

Další důležitou částí je přístup k databázím. Qt přináší vlastní rozhraní pro přístup a obsluhu databází. Nevýhodou je, že pro každý typ databáze je vyžadován jiný driver a základní verze Qt obsahuje jen driver pro SqlLite. Pro jiné databáze je drivery nutno přeložit a přidat je do Qt knihoven ručně. Aplikace využívá MySql databázi pro ukládání karet.

Pokud chceme využívat odchytávání stisku tlačítek myši, musíme implementovat privátní funkce tříd `void mousePressEvent(QMouseEvent *event);`. Tato metoda je volána vždy, když dojde ke stisku tlačítka na myši. V některých případech je vhodné zachytit i pohyb kurzoru po daném widgetu. K tomuto potřebujeme opět reimplementovat metodu `void mouseMoveEvent(QMouseEvent *event);` a dále v každém objektu, který vyžaduje odchycení pohybu, musí být toto nastavení povoleno funkcí `setMouseTracking(true);`. Informace k implementaci v Qt frameworku byly čerpány z [2] a [4].

5.1 Server

Základním stavebním prvkem serveru je třída `QTcpServer`, která má již implementovánu obsluhu připojování klientů. Uživatel jen nastaví slot pro obsluhu připojeného socketu `connect (mMtGServer, SIGNAL(newConnection()), this, SLOT(ServeNewConnection()))`, v kterém daný socket dostane slot pro zachytávání požadavků a slot, který po odpojení socketu

provede úklid. Dále pomocí metody `listen(addr, port)` začne server na dané adrese a portu naslouchat. Obsluhu více klientů současně zvládá `QTcpServer` automaticky a programátor se nemusí o nic víc starat. Odpadá tak nutnost používat vlákna nebo tvorbu nových procesů.

5.1.1 Obsluha klientů

Pro obsluhu klientů je implementován slot `void MtG_server::SendReply()`, který je volán vždy při zachycení zprávy. V obslužném slotu musíme zjistit, který socket generoval signál, a to za pomoci funkce `QTcpSocket* socket = qobject_cast <QTcpSocket*> (this->sender());`, která vrátí ukazatel na objekt, který generoval signál. Když už je znám socket, tak dojde ke zpracování požadavku a je připravena odpověď. Ta se buď odešle na právě získaný socket a nebo všem klientům, jenž jsou uloženi s loginy v hash tabulce.

5.1.2 Databáze

Server se připojuje k MySql databázi, v které jsou uloženy informace o kartách. Pro obsluhu databází je v Qt implementována třída `QSqlDatabase`. Server vytvoří objekt `QSqlDatabase mDb;`, kterému se nastaví parametry pro připojení k databázi a dojde k otevření spojení. Když přijde od klienta požadavek o zaslání informací o kartách, server vygeneruje za pomocí objektu `QSqlQuery query;` sql dotaz. Ten pak obsahuje výsledná data, která se musí zpracovat a vytvořit z nich textovou zprávu pro zaslání klientovi. Pokud není daná karta nalezena a výsledek sql dotazu je prázdný, dojde k odeslání odpovědi s chybou.

5.1.3 Vytvoření a zrušení hry

Při vytváření nové hry vzniká objekt, ve kterém bude daná hra obsluhována. Tato situace přináší pár problémů, a to především s odpojením daného socketu od hlavního objektu a připojením k nově vzniklému pro obsluhu hry.

Také pokud dojde k odpojení hráče nebo ukončení hry, musí být vygenerován signál, který hlavnímu objektu `MtG_server server;` oznámí, že hráči opouští hru. V obslužném slotu daného signálu dojde k úklidu hry a socketům, které zůstaly on-line na serveru se opět přepojí sloty pro obsluhu.

5.1.4 Třída MtG_game

Tato třída se na serveru stará o udržení stavu jedné aktuální hry. Server si vždy při vytvoření hry vygeneruje jedinečné ID, které se ji přiřadí a nastaví vlastnosti, jenž si vyžádal vytvářející hráč a hru uloží do seznamu her. Třída pak již nese informace, které jsou při připojení oponenta odeslány a hráči si tak sesychronizují nastavení hry. Pokud hráč vytváří hru zabezpečenou heslem, je uloženo v nastavení dané hry. Připojující se hráč musí také heslo zadat, jinak nedojde k přepojení do hry. Pokud je porovnaní hesel za pomocí metody `bool CheckPass(QString pass);` nevyhovující, je klientovi odesláno varovné hlášení. Tento prvek byl navrhován s ohledem na možné rozšíření, kdy server bude vést více her najednou. Velmi důležitým prvek třídy je signál `void ClientDis();`, který zachytává hlavní třída `MtG_server` a je tak upozorněna na ukončení hry a odpojení klientů od třídy `MtG_game`.

5.2 Klientská aplikace

Klientská aplikace se stará o vykonávání pravidel, přináší uživateli přívětivé ovládání aplikace, možnost náhledu na balíčky a jejich tvorbu, základní chat pro komunikaci mezi hráči, možnost založení hry a připojení k již vytvořeným hrám. Pro komunikaci se serverem je využita třída `QTcpSocket`. Touto třídou vytvoříme jeden objekt, který se bude předávat mezi okny aplikace, aby bylo možno neustále komunikovat se serverem. Aplikace se skládá z několika základních obrazovek a tříd. Uživatelské rozhraní je tvořeno sadou obrazovek. Jsou to tyto: připojení k serveru, obrazovka s nastavením a připojením ke hře, obrazovka herní plochy a také obrazovka s náhledem balíčku karet a poslední obrazovkou je rozhraní pro tvorbu balíčku. Další třídy popisují objekty reálného světa jako je karta, hráč, knihovna, hra, hřbitov a exile.

5.2.1 Reprezentace objektů

Nejdůležitějším objektem jsou herní karty, které jsou reprezentovány třídou `MtG_card`. Tato třída dědí od třídy `QWidget` a tímto se každá karta stávám widgetem, který se dle potřeb dá vykreslovat po herní ploše. Každý widget má metodu na nastavení rodiče, ve kterém se má vykreslit. Pokud ho necháme prázdný, dojde k zobrazení v novém okně. Každá karta má pro zobrazení implementovánu metodu `void MtG_card:: ShowCard(int x, int y, int w, int h,QWidget *parent)`, která přijímá souřadnice pro vykreslení, velikost karty a odkaz na widget, ve kterém dojde k vykreslení. Velmi důležitým prvkem karty je signál `CardOp(int op,int id);`, který je generován při stisku levého tlačítka na dané kartě. Signál je pak odchycen v hlavním okně a dojde k jeho zpracování.

Dále karta obsahuje všechny důležité záznamy, které mají dopad na hru a proměnné určující její stav. Především, kde se karta zrovna nachází, zda je tapnutá nebo útočící či blokující. Každá karta také obsahuje jedinečné ID, kterým je identifikována ve hře. ID čísla přiděluje server, a to při žádosti o vytvoření balíčku. Hráč odešle počet karet a ten je zaznamenám. První hráč má ID od 0 – počet karet v balíčku.Pak již hráč, jenž se připojuje ke hře, dostane od serveru maximální id, které je prozatím v dané hře přiděleno a od něj se očíslují karty druhého hráče.

Třídy popisující objekty: hra, hřbitov, exile, hráč, obsahují vždy seznam karet, které se vyskytují v daném objektu. Každý objekt obsahuje operace nad seznamem, a to přidávání, ubírání a vyhledávání. Vždy dle pozice v seznamu nebo ID čísla karty. Pokud karta neexistuje, je navrácena hodnota `NULL`. Také tyto objekty obsahují metody pro zobrazení karet, např.: hráčova ruka je neustále zobrazena.

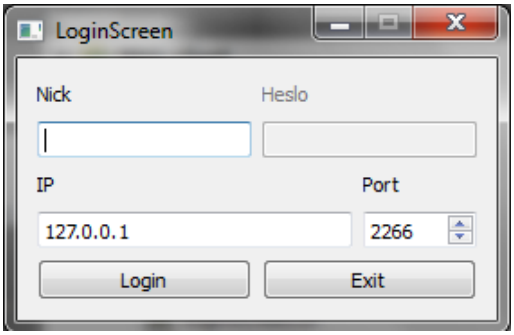
Dalším objektem je knihovna, která má navíc metody pro zamíchaní karet. První je implementací klasického míchání `void MtG_deck::SchufleDeck()`, kdy je vytvořen pomocný seznam karet. Pak dojde k vygenerování náhodného čísla od 1 do 10 a tolik karet se vždy překopíruje z jednoho seznamu do druhého. Toto se opakuje, dokud není jeden seznam prázdný. Pak se postup zopakuje v opačném pořadí, kdy jsou karty kopírovány z pomocného balíčku do základního. Metodu je nutno volat několikrát ve smyčce pro lepší zamíchání. Ideální je 10x a více. Při vytvoření nového balíčku je vždy nutné nejdříve zavolat ještě promíchání `void MtG_deck::SixPileSchufleDeck()`. V tomto míchání se vyberou všechny karty typu země z balíčku. Pak se začnou rozmíchávat karty do šesti hromádek. Vždy je daná do každé hromádky karta země, pak dvě klasické a opět země. Toto se opakuje, dokud nedojdou země a ostatní karty, pak se hromádky postupně přesunou zpět do balíčku. Díky této metodě se od sebe rozmíchají karty, které byly díky tvorbě balíčku u sebe. Knihovna

je postupně po vytvoření zamíchaná pomocí `SixPileSchufleDeck()` a pak několika násobným voláním `SchufleDeck()`. Pokud je potřeba zamíchat balíček během hry, použije se již jen několikanásobné volání metody `SchufleDeck()`.

Poslední důležitou třídou reprezentující objekt hráče ve hře, je třída `MtG_player`. Zde jsou metody pro vykreslení karet v ruce, přidávání a ubírání karet. A také metoda `int MtG_player::TakeXCard(MtG_deck *deck, int count)`, jenž usnadňuje lízání karet z balíčku. Ta přijímá odkaz na balíček a počet karet, které si má hráč dolíznout. Pokud již není daný počet karet v balíčku, metoda vrátí hodnotu 0 a dojde k ukončení hry a prohře hráče, kterému došly karty. Jestliže je karet v balíčku dost, tak se vždy určitý počet horních karet přesune do hráčovy ruky.

5.2.2 Login screen

První obrazovka po spuštění aplikace očekává od uživatele zadání informací nutných pro připojení k serveru.



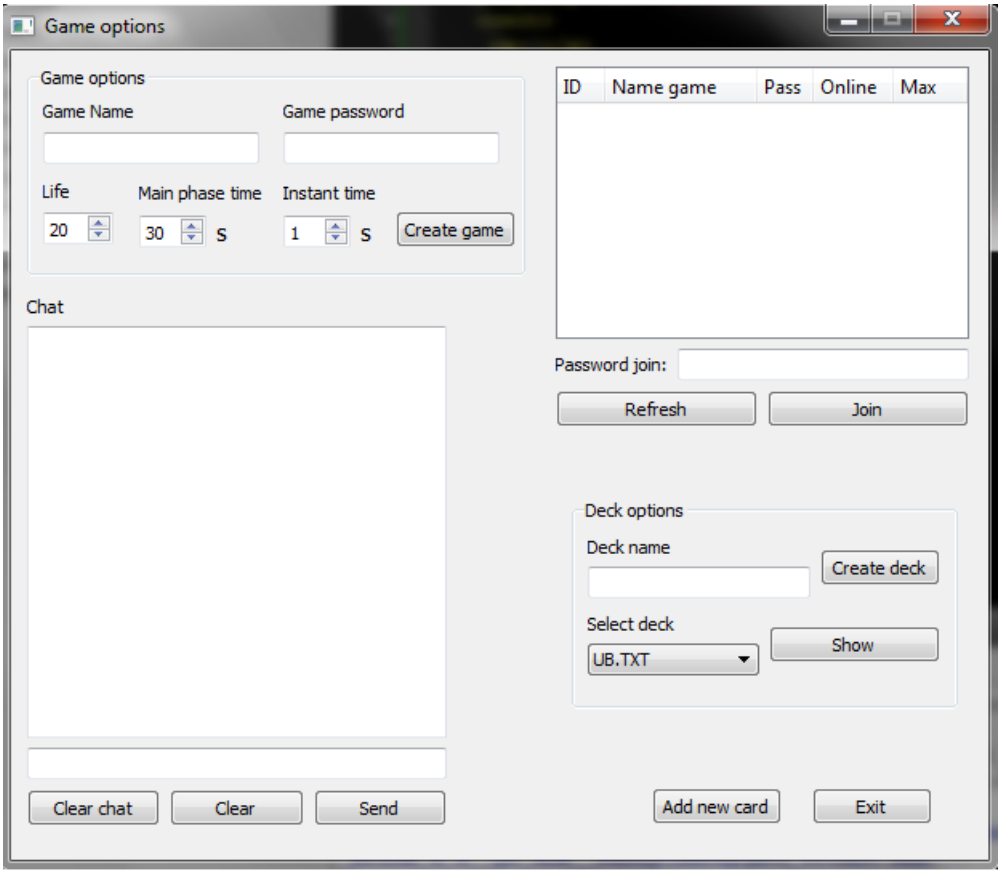
Obrázek 5.1: Obrazovka pro připojení k serveru.

Připojování k serveru je provedeno pomocí objektu `QTcpSocket* mSocket;`, kterému je předána adresa a port serveru. Po stisku tlačítka Login dojde k vytvoření socketu, připojení obslužných slotů k daným signálům (`connected()`, `disconnected()`, `error(SocketError)`, `readyRead()`) a pak k následnému připojení metodou `mSocket->connectToHost(addr, ui->spinBoxPort->value());`. Ta přijímá adresu a port, na kterém běží server. Pokud se připojení nezdaří, socket vygeneruje signál, který je zachycen hlavním oknem a obsloužen ve slotu `void LoginScrean::ConnectError(SocketError error)`. Tento slot zůstává připojen k socketu po celou dobu běhu programu a vypíše chybovou hlášku, kdykoliv dojde k chybě ve spojení. K zobrazení chyby je vytvořen objekt `QMessageBox::critical(this, tr(,,Network error‘‘), mSocket->errorString())`.

5.2.3 Options Screen

Pokud proběhne přihlášení úspěšně, dojde k zobrazení objektu `mOptions->show()` a úvodní obrazovka se skryje. V tomto okně lze komunikovat s ostatními hráči, vytvářet hry, připojit se ke hře, nahlížet na vytvořené balíčky a vytvářet nové.

Po zobrazení okna dojde k přepojení obsluhy socketu (pouze zpracování nových zpráv, ostatní signály zůstanou připojeny k původním slotům). Tímto zaručíme, že požadavky bude zachytávat nové okno a ne předchozí.



Obrázek 5.2: Obrazovka s nastavením.

5.2.4 Vytváření hry a připojení ke hře

Při vytváření nové hry se vždy otestují základní nastavení pro hru. Především zda existuje nějaký balíček karet, který je definovaný v textovém souboru. Pak se otestují a načtou data o balíčku. K tomuto slouží metoda `bool optionscrean::TestDeck(QByteArray *decktext)`. Ta předává odkazem načtená data, která se posléze odešlou na server. Pokud dojde k chybě, vypíše se varovné hlášení. Chybu může způsobit nedostatečný počet karet, neznámý soubor s balíčkem, anebo jsou zadány nevyhovující počty kusů jedné karty. Neprobíhá zde testování, zda karty existují. Toto provádí server při vytváření balíčku. Pokud vše projde v pořádku, je odeslaná žádost o zaslání informací o kartách. Server odpoví chybovým hlášením, pokud požadovaná karta neexistuje nebo není podporovaná. Pokud je vše v pořádku, dojde klientské aplikaci zpráva obsahující informace o kartách. Pak klient požádá o vytvoření hry a po potvrzení ze strany serveru, dojde k zobrazení okna, ve kterém probíhá hra.

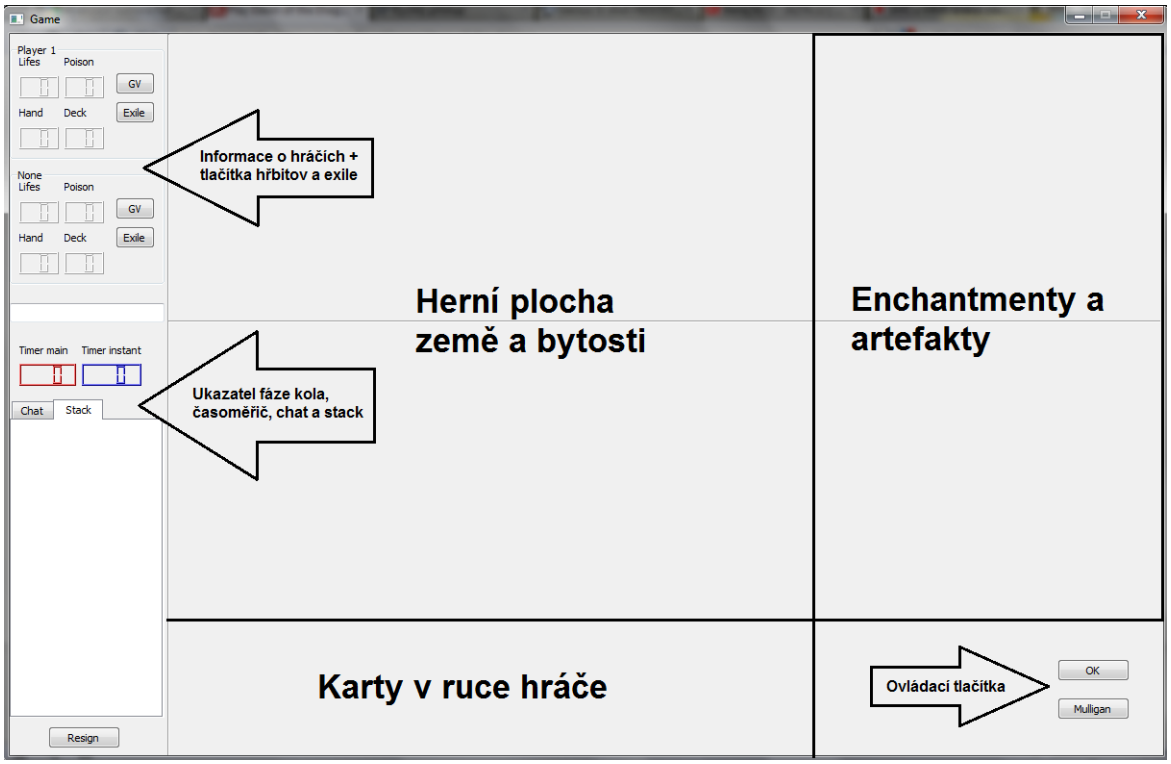
Při připojování ke hře musí být splněno několik podmínek, a to vytvoření a otestování balíčku karet. Tento postup je stejný jako u vytváření hry a zároveň musí být vybrána hra pro připojení klienta. Pak je vyslána žádost o připojení. Pokud proběhne vše v pořádku, je žádost potvrzena a klient se přepne do okna s hrou.

Po připojení obou klientů si navzájem vymění informace jako je herní přezdívka, balíčky karet a spárují si ID. Tímto se odlehčí komunikace po síti, kdy si již během hry klienti posílají, co se má provést za akci a ID karet, které se zúčastní nějaké herní akce. Nemusí se

neustále přeposílat informace o funkci karet.

5.2.5 Game screen

Po vytvoření nebo připojení k existující hře dojde k zobrazení obrazovky, ve které bude probíhat hra. Obrazovka je rozdělena do několika částí viz. obrázek 5.3.



Obrázek 5.3: Rozdělení herní plochy.

Tato implementace herní plochy se lehce odlišuje od návrhu. Byly zde z herní plochy odstraněny hřbitov a exile. Hráč je teď může najít v bočním panelu s informacemi o hře. Jsou zde tlačítka GV a Exile, které zobrazí hřbitov a exile v novém okně. Díky tomuto je na herní ploše více místa pro vykreslení enchantmentů a artefaktů.

Po připojení obou hráčů se každému zobrazí úvodní ruka, kterou může vyměnit nebo si ji ponechat a potvrdit začátek hry. Pokud oba hráči potvrdí, že jsou připraveni, server náhodně vybere hráče, který započne hru.

5.2.6 Změna fází

Pro přepínání fází kola je implementována metoda `void gamescrean::ChangePhase()`, která vždy po zavolání provede změnu fáze a odešle informaci druhému klientovi. Změna fáze se provádí ve třech případech, a to pokud vyprší čas na odehrání tahu v dané fázi. Druhým případem je situace, kdy si změnu vyžádá sám hráč stiskem tlačítka. Třetím případem jsou situace, kdy samotná hra ve fázi provede určité akce a ihned ji ukončí.

Při přepínaní fází po vypršení časovače, je využit objekt `QTimer *mTimer1sMain;`, který má připojen k signálu `timeout()` obslužný slot `EndTimerMain()`. Objekt časovače má nastaven odpočet na 1s a vždy po uplynutí této doby dojde k volání obslužného slotu, ve

kterém se odečítá z počítadla, dokud neklesne na nulovou hodnotu. Pak je volána metoda `ChangePhase()` a dojde k přepnutí fáze kola. Pro nastavení počtu sekund a spuštění časovače je využita metoda `void gamescrean::SetStartMainTimer(int time)`, jejímž parametrem je počet sekund. Po zavolání této metody dojde k nastavení počtu sekund pro odpočet, zobrazení na obrazovce a spuštění časovače.

5.2.7 Průběh souboje a rozdělení zranění

Souboj bytostí se skládá z několika fází. První fáze je deklarace útočníků. Při kliknutí na kartu, je generován signál `CardOp(int op,int id);`, který se v objektu herního okna zpracuje. Pokud je hra ve fázi deklarace útočníků a signál vygenerovala karta, která je ve hře a může útočit, stává se útočící bytostí a je volaná metoda karty `void MtG_card::ChangeAtt()`, která nastaví příznak, že se tato karta stala útočníkem. Na kartě se zobrazí písmeno A symbolizující, že byla deklarována za útočníka a aplikace si do hash tabulky jako klíčovou hodnotu uloží ID útočící karty. Pak je odeslána zpráva na server o deklaraci útočníka. Ten ji přepošle oponentovi, u kterého se zobrazí útočník.

Po skončení fáze deklarace útočníků má oponent možnost deklarovat obránce. Při výběru potenciálního obránce se za pomoci metody `bool MtG_card::IsBlocker()` zjistí, zda již bytost nebyla deklarovaná jako obránce. Pokud ano, musí být vybrána jiná bytost. V druhém případě dojde k uchování ID karty, která se chystá bránit. Následně musí blokující hráč vybrat útočníka, který bude blokován. V této části se otestuje, zda může bytost bránícího se hráče blokovat útočníka. Toto se týká karet s vlastnostmi flying a intimidate.

```
if((attacker->IsAbilit(,,flying‘‘)&&!(blocker->IsAbilit(,,flying‘‘)
||blocker->IsAbilit(,,reach‘‘)))) return;
if((attacker->IsAbilit(,,intimidate‘‘))&&!(blocker->IsType1(,,Artifact‘‘)
||attacker->EqualColorIdentity(blocker))) return;
```

Pokud je obránce přiřazen, je odeslána zpráva obsahující ID útočící bytosti a ID bytosti, která bude blokovat. Klient, který deklaroval útočníky, si v hash tabulce přiřazuje k ID útočící bytosti ID blokujících a po ukončení fáze deklarace blokujících bytostí bude rozdělovat zranění a odesílat zprávy o výsledku druhému klientovi.

Rozdělení bojového zranění probíhá v cyklu, který je ukončen po zpracování všech útočníků. Útočící bytost má v hash tabulce `QHash<int,int> mAttBlock;` uloženy ID karet, které ji blokují. Pokud není útočník blokován, je oponentovi odeslána zpráva s informací o ztrátě životů. Jestliže má útočící bytost přiřazenu minimálně jednu blokující bytost, musí být vyhodnoceno, která bytost udělí zranění jako první. K tomu slouží metoda `int MtG_card::CombatDamage(MtG_card *card)`, kterou má každá karta. Tato metoda je volána útočící kartou a jako parametr přijímá kartu blokující. Navrací celočíselnou hodnotu v intervalu <1,3>. Tato hodnota určí, která bytost udělí dříve zranění. Pokud metoda navrátí hodnotu 1, udělí si bytosti zranění navzájem,pokud hodnotu 2, první uděluje zranění bytost, která útočí a pokud není zničena, tak teprve udělí zranění bytost blokující. V posledním případě, kdy je vracena hodnota 3, udělí zranění jako první bytost, která blokuje. Pokud útočící bytost přežila udělené zranění a je ještě připravena další blokující bytost, postup se zopakuje se stejným útočníkem.

Kapitola 6

Spouštění a ovládání aplikací

Serverovou aplikaci je možné spustit s parametry nastavujícími port, na kterém server poběží. Pokud není zadán parametr portu, běží aplikace na portu 2266. Dále je možné zadat přístupové informace k MySql databázi, a to její adresu, přístupové informace a port. Pokud není zadán žádný parametr s informacemi o databázi, jsou nastaveny tyto parametry:

```
mDb.setHostName(,,localhost‘‘);
mDb.setDatabaseName(,,MtG_Cards‘‘);
mDb.setUserName(,,root‘‘);
mDb.setPassword(,,admin‘‘);
mDb.setPort(3306);
```

Server se spouští následujícím způsobem: MtG_server.exe -p <port> -dbhn <database host name> -dbdn <database name> -dbun <database user name> -dppass<database pass> -dbp <database port>. Po spuštění server začne naslouchat na daném portu a připojí se k databázi. Pokud dojde k chybě v připojení k databázi, bude vypsána chybová hláška a server ukončen.

Klientská aplikace je plně grafická s intuitivním GUI. Po spuštění aplikace je hráč vyzván, aby zadal svou přezdívku, IP adresu a port serveru. Následně se klient pokusí připojit. Pokud se připojení nezdaří, dojde k vypsání hlášení o chybě. Po připojení k serveru může uživatel vytvářet hry, připojovat se k existujícím hrám, vytvářet a nahlížet na balíčky karet.

Při vytváření nové hry je uživatel vyzván k nastavení názvu hry, který je povinný. Dále si může nastavit heslo, kterým je daná hra zabezpečena. Nastavení hesla je nepovinné. Pak hráč nastaví parametry samotné hry, jakou jsou životy, čas na odehrání fáze kola a také čas poskytnutý na reagování na některé herní události. Pokud je vše nastaveno, může hráč hru založit. Nebude-li nastaven balíček karet nebo bude obsahovat neznámé karty, bude vypsáno varovné hlášení a hra se nevytvoří.

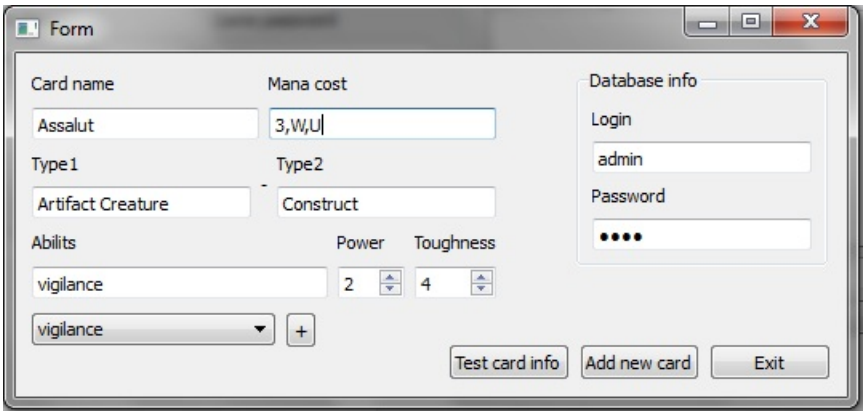
Připojení ke hře se provede výběrem hry a kliknutí na tlačítko join. Jestliže hra požaduje heslo, musí být zadáno před pokusem o připojení. Pokud tomu takto není nebo je heslo nesprávné, nedojde k připojení do hry.

Průběh hry se řídí pravidly MTG a ovládá se pouze myší. Pravým tlačítkem myši může hráč zvětšovat karty a levým vykonávat akce podle toho, jaká je zrovna fáze kola a kde se nachází karta, která má vykonat nějakou herní akci. Na obrázku 6.1 je vidět průběh hry.

Pokud chce uživatel přidat nové karty, může využít vestavěný modul, kterým přidá novou kartu do databáze na straně serveru. Při tvorbě nové karty, musí hráč zadat přístupové informace k databázi a vyplnit vlastnosti karty. Při výběru abilit může hráč použít seznam podporovaných abilit. Zde vždy jednu vybere a tlačítkem + ji přidá k vlastnostem karty.

Obrázek 6.1: Hraní hry.

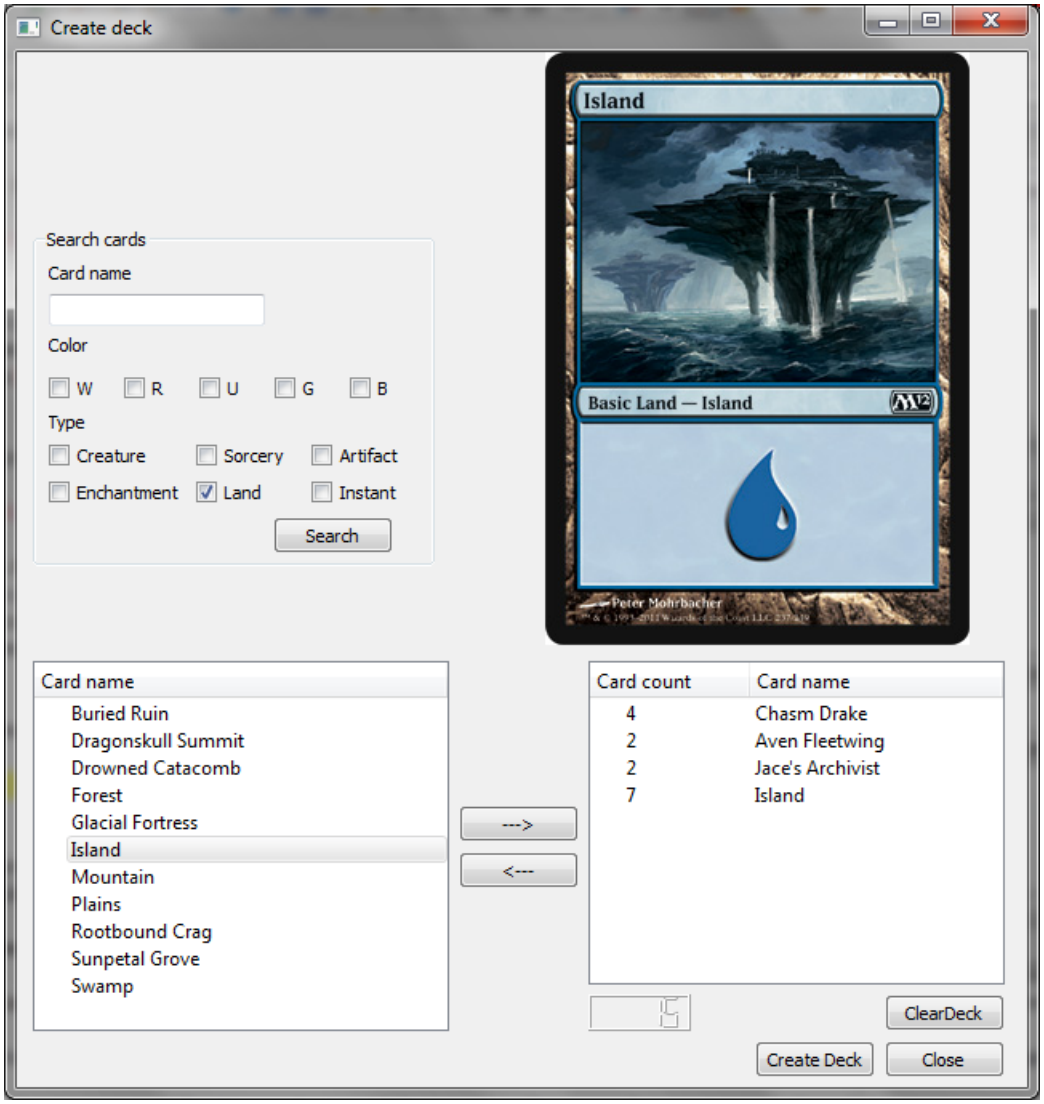
Pak již stačí kartu odeslat na server a to kliknutím na tlačítko Add new card, viz. obrázek 6.2. Tlačítkem Test card info může uživatel ověřit, zda zadal informace, které odpovídají standardnímu zápisu karty



Obrázek 6.2: Přidávání nové karty.

Pokud jsou všechny informace zadány správně, odešlou se na server a ten vrátí odpověď, zda došlo k uložení nové karty nebo k chybě. Tato informace se pak zobrazí uživateli v modálním okně.

Vytváření herní balíčku probíhá v novém okně, které je vidět na obrázku 6.3. Zde si hráč může vyhledávat podporované karty. Vyhledávat lze podle názvu, barvy nebo typu karty. Pokud je zadáno více vyhledávacích kritérií musí být všechny splněny. Po nalezení karet



Obrázek 6.3: Vytváření herního balíčku.

se zobrazí v levé tabulce seznam karet. Po kliknutí na kartu se zobrazí její velký náhled. Dvojklikem nebo pomocí tlačítka s šipkou lze přidávat karty do balíčku. Pokud chceme kartu odstranit, lze to provést obdobným způsobem. Když je balíček hotov stačí již jen kliknout na tlačítko Create deck a dojde k vytvoření balíčku. Pokud se balíček nevytvoří, je to z důvodu nedostatku karet.

Kapitola 7

Testování

Testování aplikace probíhalo během celého vývoje. V první fázi vývoje byly navrženy testy, které ověřovaly komunikaci mezi klientem a serverem. Testy byly především zaměřeny na vytváření a rušení nových her na straně serveru. V této části aplikace občas dochází k výpadkům serveru. Pokud se jeden klient připojený ke hře ukončí neočekávaným způsobem, server nezpracuje správně jeho signál o ukončení a při pokusu o uzavření hry a odpojení zbývajícího hráče, dojde k výpadku serveru. Tento problém nastával jen v určitých případech a prozatím nebyl vyřešen.

Dalším problémem, který se objevil při testování komunikace, bylo slučování zasílaných zpráv do jedné neznámé zprávy. Tento problém se vyskytoval vždy, když byla komunikace rychlá a klient či server nestačili zprávu přečíst, než byla doručena další. Po zjištění této komplikace byl lehce upraven komunikační protokol a do aplikací byly přidány části kódu, v kterých se sloučené zprávy rozdělily zpět.

Po dokončení implementace GUI probíhalo testování na několika operačních systémech Windows. Prvním problémem byly komplikace s načítáním obrázků během hry a také serverová aplikace nedokázala najít ovladače, které jsou potřebné pro připojení k databázi. Obě dvě komplikace byly vyřešeny přidáním tří dynamických knihoven do podadresářů aplikace. Dvě knihovny patří k databázi, je to její driver a následně ještě knihovna od výrobců MySql. Třetí dynamická knihovna přináší podporu pro zobrazení jpeg obrázků ve hře.

Závěrečné testování bylo zaměřeno na samotnou hru a její průběh. Testování se zúčastnili poloprofesionální hráči, ale i absolutní začátečníci. Výsledkem testování byly návrhy na úpravu vzhledu a pravidel.

Díky velmi složitým pravidlům karetní hry MTG, jejím různým možnostem a množství karet, nebylo možno hru plně otestovat. Některé chyby v aplikaci nebo v řešení pravidel bude nutné upravovat po nasazení do provozu. Rovněž vždy, když vyjde nové edice obsahující nové karty, bude potřeba program lehce upravit a opět otestovat.

Kapitola 8

Závěr

V této bakalářské práci byla úspěšně implementovaná síťová hra Magic: The Gathering. Hra dle zadaní podporuje hru dvou hráčů, tvorbu nových balíčků, přidávání nových karet. Základní verze hry podporuje jen některé karty. Ostatní se budou moci doimplementovat.

Každý hráč, který zná přístupové informace k databázi smí přidávat nové karty. Tato vlastnost aplikace je omezena, a to pouze na přidávání karet typu bytost. Tyto karty smí mít pouze jednoduché jednoslovné ability, které hra podporuje. Pokud je přidaná nová karta, budou muset uživatele také přidat obrázky, které musí mít stejný název jako nová karta.

Práce se skládá ze dvou aplikací. První je klientská část. Ta je plně grafická a přináší hráči rozhraní pro komunikaci se serverovou aplikací. Klientská aplikace řeší všechna pravidla během hry a posílá druhému hráči výsledky daných herních situací. Druhou části práce je serverová aplikace, která se pouze spustí a již nepotřebuje žádnou obsluhu. Při spouštění serveru se pouze nastaví parametry pro přístup k databázi a port na kterém má server naslouchat. Pokud je server spuštěn bez parametrů budou nastaveny implicitní. Serverová aplikace se stará o základání a rušení her, ale také udržuje spojení mezi klienty, kteří zrovna hru hrají.

Server byl implementován s ohledem na případná rozšíření, a to především na možnost vedení uživatelských účtů a jejích statistik, hraní her mezi více hráči a případnou možnost přidat i limited typů hry.

V práci byl navrhnut a aplikován komunikační protokol, který odpovídá přesným požadavkům hry. Jsou v něm dva typy zpráv. Jedny pro komunikaci se serverem, kde odpověď dostanou všichni online hráči nebo jen odesílatel zprávy. Druhým typ zpráv není na serveru zpracováván, je pouze přeposlán danému klientovi. Tyto zprávy se využívají během samotné hry, kde server pouze udržuje spojení mezi klienty.

Díky neustálému vývoji hry Magic: The Gathering je nutno aplikaci neustále udržovat aktuální. Verze, které je nyní hotova je zaměřena především na uživatelské rozhraní a obsahuje pouze pár základních karet. Další karty a nové vlastnosti je možné neustále přidávat.

Literatura

[1] Berson, A.: *Client/server architecture*. McGrawHill, 1996, iSBN 978-00-700-5664-0.

[2] Blanchette, J.: *C++ GUI programming with Qt 4*. Prentice-Hall, 2008, iSBN 0-13-235416-0.

[3] Kabelová, A.: *Velký průvodce protokoly TCP/IP a systémem DNS*. Computer Press, 2008, iSBN 978-80-251-2236-5.

[4] Molkentin, D.: *The book of Qt 4 : the art of building Qt applications*. No Starch Press, 2007, iSBN 978-1-59327-147-3.

[5] Nath, D. K.: *C Programming Essentials*. Pearson Education India, 2010, iSBN 978-81-371-2889-5.

[6] Sosinsky, B.: *Mistrovství-počítačové sítě*. Computer Press, 2010, iSBN 978-80-251-3363-7.

[7] Sprankle, M.: *Problem Solving and Programming Concepts*. Prentice Hall, 2011, iSBN 978-01-311-9459-5.

[8] Wizards of the Coast: Magic: The Gathering Rules. `http://www.wizards.com/Magic/TCG/Resources.aspx?x=magic/rules`.

Dodatek A

Obsah DVD

Na přiloženém DVD nosiči se nachází: návod na instalaci a spuštění hry, zdrojové kódy hry, samotná spustitelná aplikace a soubory nutné pro její běh. Také je přiložen SQL skript, který vytvoří a naplní databázi karet.